Hardware-in-the-Loop Evolution of a 3-bit Multiplier

Gregory V. Larchev
QSS Group, Inc.
NASA Ames Research Center, MS 269-3
Moffett Field, CA 94035-1000
glarchev@mail.arc.nasa.gov

Jason D. Lohn
Computational Sciences Division
NASA Ames Research Center, MS 269-1
Moffett Field, CA 94035-1000
jlohn@mail.arc.nasa.gov

Field Programmable Gate Arrays (FPGAs) have a number of advantages which make them particularly suitable for space applications. While operating on a spacecraft, FPGAs are susceptible to different kinds of failures. Since it is virtually impossible to replace spacecraft components in-situ, there is a clear opportunity for fault-tolerant FPGA circuits.

Evolutionary algorithm (EA) methods hold promise in their ability to search across the space of FPGA configurations for those that can function in the presence of certain types of faults. EAs are search algorithms that coarsely model Darwinian evolution and genetics to find solutions to optimization and design problems.

Real-time fault repair first became possible with the introduction of SRAM-based FPGA devices. In SRAM-based chips the number of programming cycles is unlimited. Therefore it becomes possible to restore the functionality through the repair of the compromised FPGA, a property which our algorithm explores.

Some of the main tasks space vehicles perform commonly involve Digital Signal Processing. Our motivation for evolving a multiplier circuit stems from its importance in DSP applications. Other groups in the evolvable hardware community have also recognized the importance of FIR filters and multipliers. Despite the large amount of research conducted in the field of fault tolerance and fault repair, only a few groups have studied evolution performed in physical hardware. Evolving circuits in physical hardware is a crucial step for testing fault-repair algorithms.

In our project, we focused on evolving a 3x3-bit multiplier from scratch. On an actual mission, our method would assume a dual-redundant FPGA system whereby the faulty FPGA undergoes evolution to recover its functionality while the redundant FPGA maintains proper functionality during evolution on the faulty FPGA. Thus after a fault is detected, redundancy is lost for a short period of time and then restored.

A 3x3-bit multiplier has been previously evolved in simulation by Vassilev [4]. We set up our evolutionary framework in such a way that the structure of our evolved multiplier would be similar to that of Vassilev's. The design of our multiplier is purely combinational, so that no feedback loops are allowed. The evolved multiplier can potentially take up to 48 LUTs. For our genome we use a bitstring representation. The complete chromosome is 1536 bits long. The routing for each circuit is laid out automatically by the JBits software.

The GA software we use is ECJ, a Java-based evolutionary computation and genetic programming system by Sean Luke of George Mason University. ECJ is augmented by our code for tasks like decoding individuals and calculating fitness. The GA is interfaced with Xilinx Corporation's JBits software. JBits is also used to download the circuit configuration onto the Celoxica RC1000 FPGA prototyping board, and to deliver data to and from the board. We performed approximately ten EA runs, one of which produced a 100% accurate 3-bit multiplier.

The evolved multiplier is shown in Figure 1. Using the specified evolutionary parameters, the evolution took approximately 114,000 generations. Evolving a multiplier can be thought of as the most severe case of fault repair. From our experience, repairing one or several induced faults in a circuit is significantly easier than evolving that circuit from scratch; thus, it would be reasonable to expect that actual fault repair would take significantly less time.

The main benefit of our algorithm is the fact that it tests evolved solutions on the physical FPGA, as opposed to a simulated one. This enables the algorithm to take into account the physical features of the device (faults would fall under that category), and relaxes the requirement of fault location and isolation. The overall time required for the evolutionary process to complete is still considerably longer than desired. This is because automatic routing algorithm in JBits software is extremely time consuming (according to our findings, routing operations accounted for about 90% of the overall execution time.) Future work includes investigating other hardware options to reduce our dependency on JBits software and making our evolutionary algorithm more efficient and better suited for larger, more complex problems.

References

[1] J.Lohn, G.Larchev, R.DeMara, "A Genetic Representation for Evolutionary Fault Recovery in Virtex FPGAs," in Proceedings of Evolvable Systems: From Biology to Hardware, 5th International Conf., ICES 2003, March 2003, Trondheim, Norway.

[2] J.F.Miller, M. Hartmann, "Evolving messy gates for fault tolerance: some preliminary findings," in Proceedings of the Third NASA/DOD Workshop on Evolvable Hardware, July 12-14, 2001, Long Beach, CA.

[3] A.Thompson, "Silicon Evolution," in Proceedings of Genetic Programming Conference 1996, July 28-31, 1996, Stanford, CA.

[4] V.K.Vassilev, D.Job and J.F.Miller, "Towards the Automatic Design of More Efficient Digital Circuits," in Proceedings of The Second NASA/DOD Workshop on Evolvable Hardware, July 13-15, 2000, Palo Alto, CA.

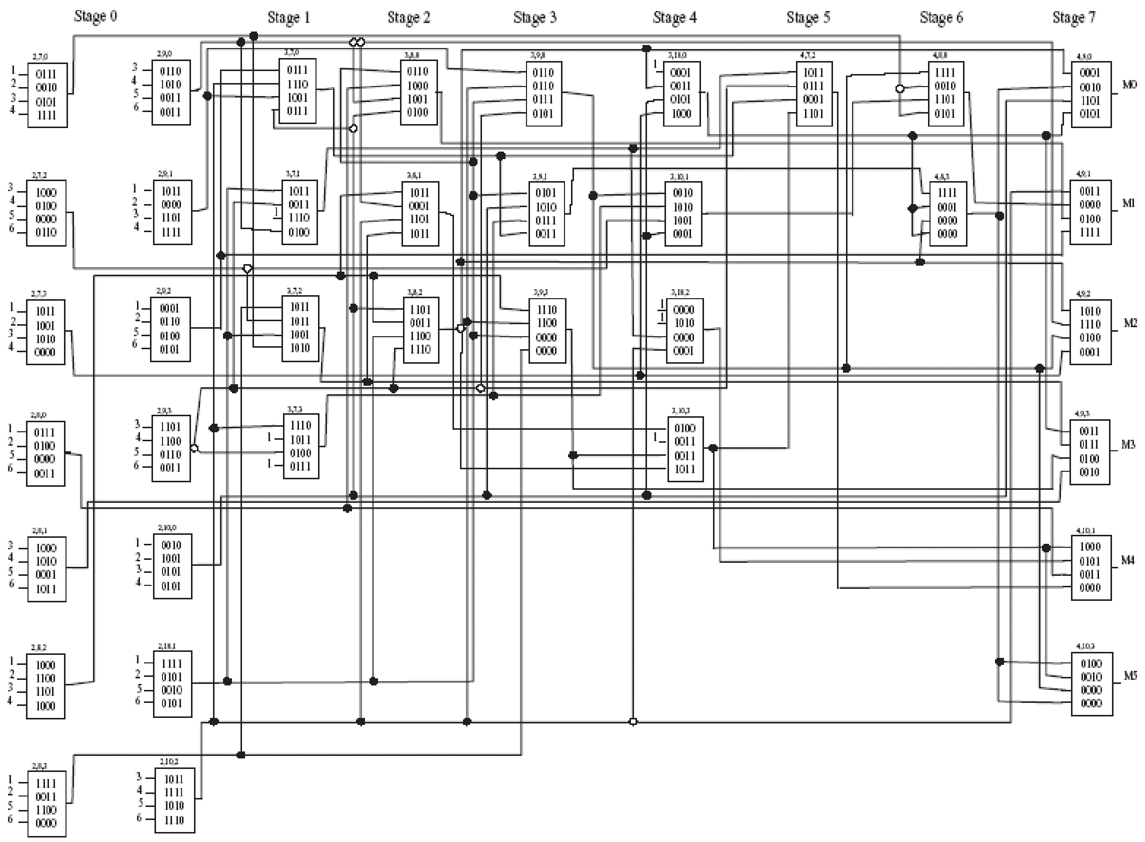

Figure 1: Evolved 3-bit multiplier configuration: 37 total LUTs in eight stages with evolved logic and connections.